A CONTEXT-BASED NAVIGATION PARADIGM FOR ACCESSING WEB DATA

Wilfried Lemahieu

K.U.Leuven - Department of Applied Economic Sciences
Naamsestraat 69
3000 Leuven, Belgium
Email: Wilfried.Lemahieu@econ.kuleuven.ac.be

KEYWORDS

Hypermedia, *Context-based navigation*, *Guided tours*, *Dynamic link generation*

ABSTRACT

This paper presents a context-based navigation paradigm, so as to overcome the phenomenon of user disorientation in a Web environment. Conventional navigation along static links is complemented by run-time generated guided tours, which are derived dynamically from the context of a user's information requirements. The result is a two-dimensional navigation paradigm, which reconciles complete navigational freedom and flexibility with a measure of linear guidance. Consequently, orientation is improved through reduced cognitive overhead and an increased sense of document coherence.

1. INTRODUCTION

The *MESH* hypermedia framework as deployed in [1] proposes a structured approach to both data modeling and navigation, so as to overcome two crucial problems in the field: *poor maintainability and user disorientation*. To start with, this paper portrays the predicaments surrounding hypermedia navigation. Next comes a brief overview of *MESH*'s data model and implementation framework. The body of the paper discusses *MESH*'s *context-based navigation paradigm* in detail. A last section makes comparisons to related work and formulates conclusions.

2. THE HYPERMEDIA NAVIGATION PROBLEM

2.1 Navigation versus querying

Hypermedia systems stand out amid other information systems in that data access is accomplished through *navigation*, rather than querying. Their appeal is based upon the ability to store complex, cross-referenced bodies of information as a network of *nodes* and *links*, which can be explored according to the user's personal preferences. Therefore, hypermedia data retrieval embraces a notion of *location*. Data accessibility depends on a user's position in the network, denoted as the *current node*. Manipulation of this position gradually reveals links to related information. The environment that really brought hypermedia to the public eye is undoubtedly the *World Wide Web*, promoting the hypertext paradigm as the primary access mode to all nodes, implemented as *HTML pages*, on Internet-connected networks around the world.

However, efficient hypermedia data retrieval requires a search mechanism to complement navigational access. Therefore, query-based applications (e.g. search engines) have established themselves in the *Web* as well. However, *navigation* remains an utterly valuable asset to further explore the available information, once an initial *starting point* has been obtained by means of a query.

2.2 Navigation and (dis)orientation

As a downside, the explorative, non-linear nature of hypermedia navigation imposes a heavy processing load upon the end user, referred to as *cognitive overhead*, particularly in such an immense environment as the *Web*. The stringent problem of cognitive overhead effecting into user disorientation and losing one's chain of thought is known as the 'lost in hyperspace' phenomenon [2].

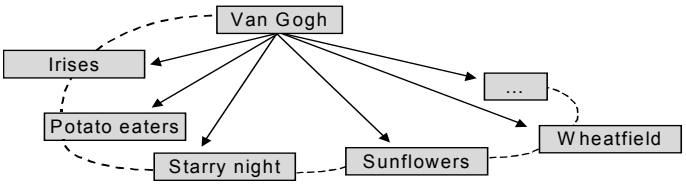
Recently, a distinction has been made between *hyperspace* and *conceptual space*. The former refers to the hypermedia structure itself, whereas the latter involves the actual concepts and interrelations represented in the hypermedia system. A *close correlation between hyperspace and conceptual space* is claimed to significantly advance comprehension and orientation. In addition, [3] formulates design principles such as the use of typed links, higher-order information units, a stable screen layout and visual cues, to *increase document coherence* and *reduce cognitive overhead*. These principles are realized in *MESH*'s data model, which is briefly reviewed in section 3.1.

With respect to navigation, the same authors suggest to provide information about the *context* in which a node is displayed. This conveys a sense of continuity across separate nodes and reduces the impression of information fragmentation. Moreover, navigation facilities are to cover aspects of *direction and distance* and should offer the reader maximal support to identify his *current position* within the hypermedia structure. Also*, selection of the next navigation step* should be made as easy as possible. The latter principles provide the fundaments to *MESH*'s navigation paradigm. Another key ingredient is the notion of *guided tours*, as introduced below.

2.3 Linearity and guided tours

To highlight the advantages of hypermedia navigation, comparisons are often made to books. Books are said to be *linear* information systems; their pages are organized uni-dimensionally, in a fixed order. Hypertext offers the possibility to break through this linear constraint and organize data in more complex structures, to be accessed following different possible paths, depending on the user's preferences and interests.

Cognitive overhead, however, is significantly lower in a linear structure, be it at the cost of navigational freedom. Linearity provides a leading thread that facilitates orientation and prevents the reader from getting lost. Therefore, linearity is re-established by means of so-called *guided tours*, chaining together all nodes pertaining to a common subject with *forward*/*backward* links. E.g. the typical hypermedia links (represented as arrows) between **Van Gogh** and each of his **paintings** can be complemented by a *guided tour* (represented as dotted lines) along these **paintings**.



Unfortunately, such hard-coded guided tours have proven to be inflexible and difficult to maintain. Moreover, they introduce a measure of redundancy into the hyperbase, as a guided tour typically reflects a communal property among its participating nodes. However, the property of '*being painted by the same artist*' is already established within the respective links from each **painting** to its **artist**. Thus, it would be possible to infer this knowledge and generate such guided tour *at run-time*, without burdening hyperbase maintainability.

MESH's improved navigation paradigm, referred to as *context-based navigation*, reconciles navigational freedom with the ease of linear navigation. In this manner, the concept of *at run-time generated guided tours* offers a linear path throughout (part of) the hyperbase to reduce cognitive overhead and user disorientation.

3. THE MESH HYPERMEDIA FRAMEWORK

MESH is an acronym for *Maintainable*, *End user friendly*, *Structured Hypermedia*. Elements from both *entity-relationship* and *object-oriented* modeling, along with proprietary hypermedia concepts, are combined into a

framework that should improve both *maintainability* and end user *orientation*. This section briefly discusses *MESH*'s data model and implementation framework. The body of this paper is dedicated to its *navigation paradigm*.

3.1 The MESH data model

The data model is looked upon as an inheritance hierarchy of *node types*, bestowing nodes representing similar real-world entities with a uniform layout. The latter is abstracted on type level in *layout templates*.

So as to characterize node interaction, each node type is equipped with a set of *link types*. These classify the links according to the semantic interpretation of the relations they embody. The properties of a link type are its *domain*, *minimum cardinality* (optional/mandatory), *maximum cardinality* (unique/non-unique) and *inverse link type*. Semantics attributed within the data model permit automated type checking and integrity constraints. Moreover, node design is greatly facilitated by the abstract-level node and link type definitions, cardinality constraints and layout templates. These can be inherited and refined, but their constraints may never be weaker than the ones imposed at a higher level.

Whereas the data model as depicted thus far is inherently *static*, with each node being an instance of exactly one "most specific type", which remains unchanged during the whole of the node's lifetime, the *aspect* construct offers a means to overcome modeling restrictions imposed by a rigid subtyping hierarchy. *Aspect descriptors* provide for additional node classification criteria, such that affiliated properties (i.e. both specific layout and link types) can be bundled into *aspects*, attributed to *temporary* and *non-disjoint* sets of nodes. At run-time, a node's features can be altered through manipulation of its aspect descriptor *values*.

Nevertheless, the inheritance hierarchy remains the backbone of the hypermedia model, with aspects themselves being defined as a node type property that can be attributed within the hierarchy and be inherited and refined at lower levels. In this respect, aspects can be seen as node type building blocks that describe one specific "aspect" of a node.

As a last issue, link subtyping is further elaborated upon. In contrast to standard O.O. practice, link types are full-fledged abstract types and can be equally subject to subtyping. The latter can be either in parallel with or independent of node classification. Link types are deemed extremely important, as they not only enforce semantic constraints but also *interface* between nodes, such that these can be coded and updated independently of one another. Moreover, they define navigational actions on an abstract level, which provides the key to *context-based navigation*.

3.2 A generic application framework

The *information content* and *navigation structure* of the nodes are separated and stored independently. The resulting system consists of three types of components: the *nodes*, the *linkbase/repository* and the *hyperbase engine*. In [1], a platform-independent implementation framework was provided, but all subsequent prototyping is explicitly targeted at a *Web* environment.

A node can be defined as a static page or a dynamic object, using either HTML or XML. Its internal content is shielded from the outside world by the indirection of link types playing the role of a node's interface. Optionally, it can be endowed with the intelligence to tune its reaction to the *context* in which it is accessed. Indeed, by integrating a node type's set of attributed link types as a parameter in its layout template's presentation routines, the multimedia objects that are most relevant to this particular link type can be made current upon node access, hence the so-called *context-sensitive visualization* principle.

Since a node is not specified as a necessarily searchable object, linkage information cannot be embedded within a node's body. Links, as well as meta data about node types, link types, aspect descriptors and aspects are captured within a searchable *linkbase/repository* to provide the necessary information pertaining to the underlying hypermedia model, both at design time and at run-time. This repository is implemented in a relational database environment. Only here, references to physical node addresses are stored, these are never to be embedded in a node's body. All external references are to be made through location independent *node ID*'s.

The *hyperbase engine* is conceived as a server-side application that accepts link (type) selections from the current node, retrieves the correct destination node, keeps track of session information and provides facilities for generating maps and overviews. Since all relevant linkage and meta information is stored in the relational DBMS, the hyperbase

engine can access this information by means of simple, pre-defined and parameterized *database queries*, i.e. without the need for searching through *node content*.

4. MESH'S CONTEXT-BASED NAVIGATION PARADIGM

4.1 A guided tour as derived from the current context

In conventional hypermedia applications, the *current node* is the only variable that determines which information is accessible at a given moment; navigation is only possible to nodes that are linked to this current node. Its value changes with each navigation step as it represents the immediate focus of the user's attention.

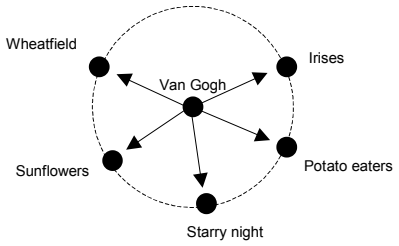
MESH introduces the *current context* as a second, longer-term variable that 'glues' the various visited nodes together and provides a background about which common theme is being explored. The current context is defined as the combination of a *context node* and a *context link type*. The context node represents the subject around which the user's broader information requirements 'circle'. The nature of the relationship involved is depicted by the context link type.

A guided tour derives from the current context. Therefore, *MESH* discriminates between *direct* and *indirect* links. A direct link represents a lasting relation between two nodes. Direct links are typed and reflect the underlying conceptual data model. Because they are permanent and context-independent, they are stored explicitly into the hyperbase and are always valid. E.g. the node **Sunflowers** is directly linked to the **Van Gogh** node.

An *indirect* link between two nodes indicates that they share relevancy to a common third node. The latter denotes the *context* within which the indirect link is valid. As indirect links not only reflect the data model, but also depend on a run-time variable, the *current context*, they cannot be stored within the hyperbase. They are to be created *dynamically* at run-time, as inferred from a particular context. E.g. an indirect link between **Sunflowers** and **Wheatfield** is only relevant when exploring information related to **Van Gogh**.

A *guided tour* is defined as a path of *indirect* links along all nodes relevant to the current context. These nodes are directly linked to the context node (through instances of the context link type) and indirectly to their predecessor and successor in the tour. As they are chained into a linear structure, a logical order should be devised in which the subsequent tour nodes can be presented to the user. The most obvious criterion is in alphabetical order of a *node descriptor* field. More powerful alternatives are discussed in [19].

E.g. the context **Van Gogh**.***has-painted*** yields a guided tour among the nodes {**Irises**, **Potato eaters**, **Starry night**, **Sunflowers, Wheatfield,** …} with **Van Gogh** as the *context node* and ***has-painted*** as the *context link type*.

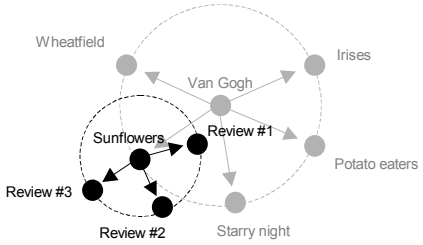


Current context: **Van Gogh.*has-painted***

Note that the discrepancy between *guided tour* and *context* can be compared to the traditional duality in representing a circle either through the points on its *circumference*, or through its *center* and a *radius*. Guided tours are not stored within the hyperbase as *an enumeration of participating nodes*, but are calculated at run-time from the *current context*. Although sequential by nature, such tours do not restrict the user's navigational freedom, as long as sufficient flexibility is offered in choosing which tour to follow. The linearity lies in 'following' the tour. The freedom lies in starting one.

4.2 Nested tours

Contexts, and consequently guided tours, can exist in 'layers'. E.g. from the current tour's current node **Sunflowers**, one is able to start a new tour, **Sunflowers.*reviews***, nested in the former. A new context emanates, with **Sunflowers** as the new context node, resulting in new indirect links. When this most recent tour has been finished, it is possible to restore the former context and resume the first tour, of which **Sunflowers** again becomes the current node.

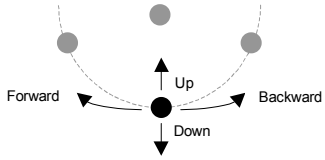


Current context: **Sunflowers.*reviews***

As such, it is possible to 'delve' into a subject and have multiple *open* tours, nested within one another, where the context node of one tour is the current node of the tour it is nested in. Navigation along indirect links is invariably carried out within the "deepest", i.e. most recently started tour. Continuing a tour on a higher level is only possible if all tours on a lower level have been either completed or disbanded.

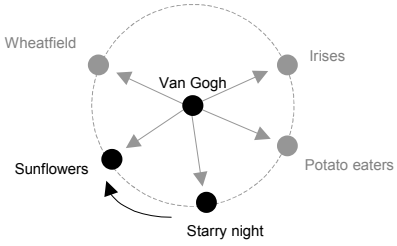
4.3 Navigational actions

Navigational actions can be classified according to two dimensions. First, there is moving *forward* and *backward* within the current tour, along indirect links. Second, and orthogonal to this, there is the option of moving *up* or *down* along direct links, closer to or further away from the session's starting point. Additionally, one can distinguish between actions that change the current context and actions that only influence the current node.



4.3.1 Moving forward/backward within the current tour

Moving forward or backward in a guided tour along indirect links, results in the node following/preceding the current node being accessed to become the new current node. The current context is unaffected.



Current context: **Van Gogh.*has-painted***
Current node: **Starry night ⇒ Sunflowers**

Indirect links are not explicitly *anchored*. A node should only return a *move forward* or *move backward* command, independently of the current tour. It's the hyperbase engine's task to map this to the correct destination node.

4.3.2 Moving down

Moving down implies an action of 'digging deeper' into the subject matter, moving away from the starting point. This is accomplished through selection from the current node of either a direct link type or link instance. In the case of a *unique* destination node, the result is the latter node being accessed. In the case of a *set* of destination nodes, the outcome is a new nested tour being started.

4.3.2.1 Selection of a link instance

A link *instance* is defined as a (*source node*, *destination node*) tuple. Selection of a link instance l from the source node n_s results in its destination node n_d being accessed:

$n_s.l := \{n_d \mid l = (n_s, n_d)\}$

As a link instance corresponds to a single tuple, the outcome is of course a singleton, hence a *single node*, being accessed. E.g. selection of the link (**Sunflowers, National Gallery**) from the current node **Sunflowers**, induces an access to the node **National Gallery**:

Sunflowers.(Sunflowers, National Gallery) := {**National Gallery**}

Anchoring link *instances* is to be avoided where possible since such anchors cannot be managed on an aggregate (i.e. node *type*) level, which is unfavorable from a maintenance point of view. Luckily, thanks to the use of typed links, along with laid down cardinality restrictions, references to link instances can often be replaced by a single reference to a link *type*.

4.3.2.2 Selection of a link type

Indeed, *MESH* aggregating single *link instances* into *link types*, yields the opportunity of anchoring and consequently selecting a *complete link type* from a given source node. Selection of a link type L from a source node n_s yields a set of all destination nodes n_d of tuples representing link instances of L with n_s as the source node, i.e. all nodes that are linked to the current node by the selected link type:

$n_s.L := \{n_d \mid (n_s, n_d) \in L\}$

Depending on maximum cardinality of the link type, the result may either still be a single node access or a change in context.

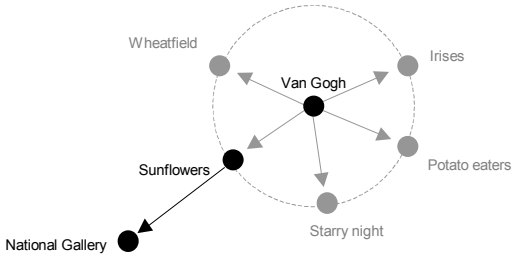
4.3.2.2.1 Selection of a unique link type

In the case where a *unique* link type is selected, the resulting "set" of nodes will always be a singleton, since a unique link type can have only one instance per source node. E.g. selecting the link type ***exhibited-in*** from the node **Sunflowers** has the singleton {**National Gallery**} as a result.

Sunflowers.*exhibited-in* := {National Gallery}

Note that, for a unique link type, anchoring link *type* or link *instance* are equivalent; selection of a single link instance or the complete link type leads to the same single destination node for any source node. Therefore, such links should only be anchored on *type* level, which offers a bonus in maintenance, as such anchor needn't be replaced when the destination node is altered for a given source node. Moreover, as even static HTML pages are parsed in *MESH*, a link type's anchor can still be tailored at run-time to reflect the actual destination node's name.

The result of selecting a *unique link type* (or a *link instance*) is a node directly linked to the current node becoming the new current node. Other parameters are unaffected.



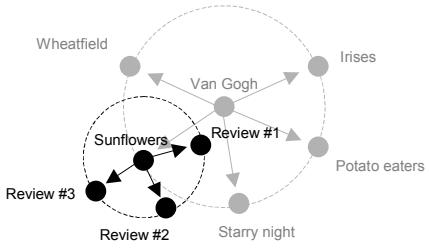
Current context: **Van Gogh.*has-painted***
Current node: **Sunflowers ⇒ National Gallery**

4.3.2.2.2 Selection of a non-unique link type

If the selected link type is *non-unique*, the resulting set may contain multiple destination nodes. E.g. with **Sunflowers** as the current node, selection of the link type ***reviews*** generates a *collection* of nodes to-be-accessed:

Sunflowers.*reviews* := {review#1, review#2, review#3, …}

The current node **Sunflowers** is denoted as the new context node. The non-unique link type ***reviews*** defines the context link type, which yields a new *nested* tour: **Sunflowers.*reviews***. The first review is accessed to become the new current node. Such *context change* reflects the user's decision to concentrate on the current node as a new topic of interest. All indirect links are destroyed and redefined around this new context.



Current context: **Van Gogh.*has-painted* ⇒ Sunflowers.*reviews***
Current node: **Sunflowers ⇒ Review#1**

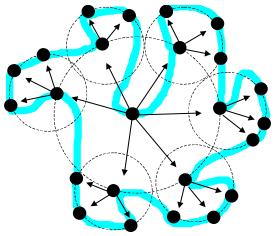
4.3.3 Moving up

Moving up reverses the latest *move down* action. If the latter involved a *unique link type or link instance selection*, the effect of this link selection is cancelled. If the latest link type selected was *non-unique*, the move up action results in the reestablishment of the previous context and the cancellation of the tour generated through this most recent link type selection. The previous context's context node and indirect links are restored. The most recent context node becomes the current node.

4.4 Issuing navigational actions on the level of a *complete tour*

The practice of node and link typing allows for casting navigational actions to a whole *class* of nodes, regardless of the actual instance they are applied to. In this way, selections of link *types* that exist at a sufficiently high level of abstraction can be imposed upon every single node belonging to a tour. E.g. in the context of **Van Gogh.*has-painted***, a **painting**#x**.*reviews*** selection can be issued once on *tour* level, with additional (nested) tours being generated automatically for each node participating in the **Van Gogh.*has-painted*** tour.

If these tours in their turn include navigational actions on type level, a complex navigation pattern results, which can be several levels deep. Again, *forward* and *backward* links always apply to the current tour, i.e. to the open tour at the 'deepest' level.

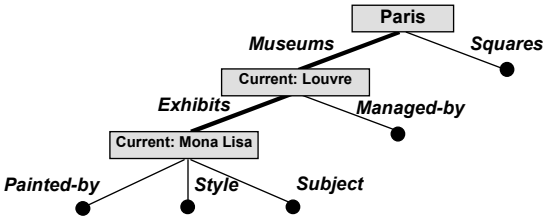


4.5 A context-based history list

MESH exploits a *context-based history list* to account for current tour positions in open tours and to keep track of navigational actions on tour level, so as to aptly impose such actions upon the successive nodes in each newly started tour.

The sequence of navigational actions to be cast upon the respective layers of guided tours can be depicted in a *tree*, with each branch representing an abstract navigational action, i.e. a *link type selection*. Each node in the tree symbolizes *the collection of hypermedia nodes accessed through the navigational action represented by the incoming branch*. The *outgoing* branches each stand for a single navigational action, to be applied to all hypermedia nodes in the collection represented by the tree node. The root node denotes the session's *starting point*.

E.g. with **Paris** as the starting point, the link types ***museums*** and ***squares*** are selected. For each node in the **Paris.*museums*** tour, a nested tour **museum**#x.***exhibits*** is generated and the destination of the unique link type ***managed-by*** is visited. For all **paintings** belonging to one of the **museum**#x.***exhibits*** tours, the respective unique link types ***painted-by***, ***style*** and ***subject*** are issued. There are two nested open tours, with **Louvre** and **Mona Lisa** as respective current nodes.



All information about a complete navigation session can be represented in a compact fashion, based solely on high-level *navigational actions* and currency indications of open tours, without every single *node* participating in the various tours being retained. Consequently, such *context-based* history list will be much shorter and yet more powerful than its traditional counterpart. Indeed, it offers the ability of bookmarking not only single nodes, but of complete navigational situations, allowing to swiftly recapitulate results of earlier exploration. These can be regenerated by applying the navigational actions represented in the tree according to a *depth-first* strategy, with the desired sequence of nodes as a result:

{Paris,
  {Louvre,
    {Helena Fourment with a Carriage, Rubens, Baroque, Helena Fourment,
     Mona Lisa, Da Vinci, Renaissance, Lisa del Giaconda,
     …},
    The French Ministry of Culture,
    Musée d'Orsay,
    {Dancing Lesson, Degas, Impressionism, A dance class around 1874,
     Bedroom at Arles, Van Gogh, Expressionism, Residence in Arles,
     …},
     The city of Paris,
   …},
  {Place de la Bastille, Place Charles-de-Gaulle, Place de la Concorde, … }}

Note that the tree representation is not only useful for *storing* a navigational situation, but also allows for *visualizing* a user's current situation in a highly compact manner. In this respect, the *move up* and *move down* actions indeed correspond to moving up or down in the graph.

5. CONCLUSIONS

5.1 Benefits of *MESH*'s hypermedia approach

MESH's full O.O. based data modeling paradigm should allow for hypermedia maintenance capabilities to equal their database counterpart; with unique object identifiers, monitoring of integrity, consistency and completeness checking, efficient querying and a clean separation between authoring *content* and *physical hyperbase maintenance*. *MESH* formulates specific rules for inheriting and overriding layout and link type properties, taking into account the added complexity of plural (possibly overlapping and/or temporal) node classifications. Links are treated as first-class objects, with link types being able to be subject to multiple specializations themselves, not necessarily in parallel with node subtyping. Authoring is greatly facilitated by O.O. features such as inheritance and overriding, class properties and layout templates that allow for high-level specification and lower-level refinement of node properties. Also, the meta information present in the hyperbase enables the automated suggestion of appropriate links. Finally, it is clear that a model-based approach in general facilitates information sharing, reuse, development in parallel, etc.

As to the *end user*, apart from the obvious benefit of a well-maintained hyperbase, *typed links* should permit a better comprehension of the semantic relations between information objects. The use of *higher-order* information units and the representation of collections of nodes as (source node, link type) combinations, induces a stronger sense of structure. A *node typing hierarchy* with consistent layout and user interface features, reflecting similarities between nodes, is to further increase the user's ability to grasp this underlying data structure. The abundance of meta-information as node, aspect and link types allows for enriching *maps* and *overviews* with concepts of varying granularity. *MESH*'s context-based navigation paradigm implements the design principles referred to in section 2.2 by means of dynamic linear paths throughout the information space, so as to reduce cognitive overhead. A sense of *direction and position* is induced, both *within* a single tour and orthogonally to the latter. In this respect, the comprehensive, tree-shaped history list representation imposes a flexible *hierarchical view* upon the hyperbase structure, which can be visualized as an *adaptive map*. Through the specification of navigational actions *on tour level*, complex navigation patterns can be applied to all nodes in a tour without additional effort. The *context* concept, as a representation of what various nodes have in common, also positively influences *document coherence*, especially in combination with *context sensitive node visualization.* A final benefit is the ability to *bookmark a complete navigational situation* in an utterly compact manner, with the possibility of it being resumed later on, from the exact point where it was left.

5.2 A comparison to related work

Other hypermedia approaches such as *RMM* [4] and *OOHDM* [5] also feature specific topologies such as *guided tours*, *indexes* etc. A fundamental difference is that these are conceived as explicit *design components*, requiring author input for query definitions, node collections and forward/backward links. In *MESH*, only *direct* links are to be explicitly authored. Maintenance is accomplished rather effortlessly, as they are stored separately in a relational database. Moreover, they can be anchored on *type* level, e.g. a *painted-by* anchor being defined only once for the node type **painting**, instead of for each **painting** instance. Neither guided tours nor indexes require any maintenance or design effort, as the author is not even engaged in their realization. They are generated at run-time upon user request. For that purpose, the appropriate queries are inferred automatically by the hyperbase engine, with the context node and context link type as input parameters.

Set-based hypermedia paradigms such as *CHM* [6], the *HM-Data Model* [7] and *Hyper-G* [8] equally provide inherent support for navigation in two orthogonal planes; *inside a collection* and *across collection boundaries*. Their *current container* and *current member* concepts are comparable to *MESH*'s *current context* and *current node* respectively. A drawback, however, is that they *do* severely limit navigational freedom. Moreover, they lack a firm underlying data model with typed node interrelations. Likewise, the opportunity of defining abstract navigational actions on tour level is a feature that is exclusive to *MESH*.

While featuring dynamic node content as well as dynamic links, *MESH* intentionally differs from typical *adaptive hypermedia systems* as reviewed in [9]. In so-called "direct guidance" systems, adaptation is based on a *user model* to provide for tailored guided tours. E.g. *Webwatcher* [10] suggests interesting links based on experience with previous users' browsing behavior and a set of keywords that has been provided at the beginning of the current user's session. One is "guided" along potentially relevant pages, with the system actually taking over navigation control, at least partially. On the other hand, *MESH*'s dynamism is targeted at providing a structured perspective upon the information space, supporting the user in efficiently pursuing his interests, rather than restricting his freedom. A guided tour of *indirect* links is invariably the result of a *direct* link type selection by the user: the initiative fully resides with the latter.

Therefore, adaptive hypermedia techniques could be seen as a complement to *MESH*, rather than a substitute. Indeed, especially in large hyperbases, with an oversupply of outgoing link anchors for a given source node, it could be required to impose dynamic behavior upon these *direct* links as well. A user profile could then be applied to highlight the most relevant direct link (type) anchors and hide irrelevant ones. Hence, whereas *MESH*'s current dynamism entails the generation of guided tours *in accordance with the user's actions*, the latter could be assisted by adaptive techniques *suggesting which action to take*. A related ongoing research issue is the possibility of the hypermedia system taking the initiative of reorganizing and optimizing the nested context structure emanating from the user's actions. In this respect, *MESH*'s rich modeling abstractions with typed nodes, links and aspects, as well as its context notion, could provide valuable semantics as additional input to existing adaptation techniques.

REFERENCES

1. W. Lemahieu 1999, Improved Navigation and Maintenance through an Object-Oriented Approach to Hypermedia Modelling, *Doctoral dissertation (unpublished)*, *Katholieke Universiteit Leuven*, Leuven
2. A. Cockburn and S. Jones 1996, Which way now? Analyzing and easing inadequacies in WWW navigation, *International Journal of Human-Computer Studies No. 45*
3. M. Thüring, J. Hannemann and J. Haake 1995, Hypermedia and Cognition: Designing for comprehension, *Commun. ACM Vol. 38*, No. 8
4. T. Isakowitz, A. Kamis and M. Koufaris 1998, The Extended RMM Methodology for Web Publishing, *Working Paper IS-98-18, Center for Research on Information Systems* (Currently under review at ACM Trans. Inf. Syst.)
5. D. Schwabe and G. Rossi 1998, Developing Hypermedia Applications using OOHDM, *Proceedings of the ninth ACM Conference on Hypertext (Hypertext '98)*, Pittsburgh
6. E. Duval, H. Olivié and N. Scherbakov 1995, Contained Hypermedia, *Journal of Universal Computer Science, Vol. 1*, No. 10
7. P. Srinivasan 1995, Incorporating Intelligent Navigational Techniques to Hypermedia, *Proceedings of LAIR-MIPRO '95*, Opatija
8. K. Andrews, F. Kappe and H. Maurer 1995, The Hyper-G Network Information System, *Journal of Universal Computer Science, Vol. 1*, No. 4
9. P. Brusilovsky 1996, Methods and techniques of adaptive hypermedia, *User Modeling and User-Adapted Interaction Vol. 6*, No. 2-3
10. T. Joachims, D. Freitag and T. Mitchell 1996, Webwatcher: A Tour Guide for the World Wide Web, *CMU research report*